

# Technical Cybersecurity

readelf

# Reading ELF formatted files

## READELF V. OBJDUMP

---

- Readelf: displays information on ELF formatted files
- Objdump: displays information on object files

## SIMILAR BUT DIFFERENT

---

- Dump headers, symbols, relocations with both
- objdump: disassembly, source, debugging info
- readelf: section details, segments, sections

## WHY SIMILAR?

---

- ELF files are object files, and object files (on linux) are ELF files

# When to use which?

## READELF

---

- ▶ Sections and segments
- ▶ Symbols and function names
- ▶ Entry points

## OBJDUMP

---

- ▶ Dissassembly
- ▶ ...anything code related

**Many tools show same information in different ways**

# What will it show?

## ELF FILE HEADERS

---

- ▶ Magic: ELF files have the bytes 7f 45 4c 46 (7f 'ELF') at the start of the file
- ▶ Type, processor type, entry point

```
cclamb@ubuntu:~/Work/abi-playground $ readelf -h f2
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                               ELF64
  Data:                                   2's complement, little
  Version:                             1 (current)
  OS/ABI:                               UNIX - System V
  ABI Version:                         0
  Type:                                 EXEC (Executable file)
  Machine:                              Advanced Micro Device
  Version:                              0x1
  Entry point address:                  0x4003b0
  Start of program headers:            64 (bytes into file)
  Start of section headers:            7160 (bytes into file)
  Flags:                                0x0
  Size of this header:                  64 (bytes)
  Size of program headers:              56 (bytes)
  Number of program headers:            9
  Size of section headers:              64 (bytes)
  Number of section headers:            32
  Section header string table index:    31
cclamb@ubuntu:~/Work/abi-playground $
```

# Symbols

## FUNCTIONS

---

- **\$ readelf -s f2**
  - You can see our functions
  - ...and a bunch the compiler inserted for us
    - `_start`, `_init`, etc.
  - FUNC -> function
  - OBJECT, NOTYPE, FILE

```
0 SECTION LOCAL DEFAULT 28
0 FILE LOCAL DEFAULT ABS crtstuff.c
0 FUNC LOCAL DEFAULT 11 deregister_tm_
0 FUNC LOCAL DEFAULT 11 register_tm_cl
0 FUNC LOCAL DEFAULT 11 __do_global_dt
1 OBJECT LOCAL DEFAULT 22 completed.7696
0 OBJECT LOCAL DEFAULT 17 __do_global_dt
0 FUNC LOCAL DEFAULT 11 frame_dummy
0 OBJECT LOCAL DEFAULT 16 __frame_dummy_
0 FILE LOCAL DEFAULT ABS function2.c
0 FILE LOCAL DEFAULT ABS crtstuff.c
0 OBJECT LOCAL DEFAULT 15 __FRAME_END__
0 FILE LOCAL DEFAULT ABS
0 NOTYPE LOCAL DEFAULT 16 __init_array_e
0 OBJECT LOCAL DEFAULT 18 _DYNAMIC
0 NOTYPE LOCAL DEFAULT 16 __init_array_s
0 NOTYPE LOCAL DEFAULT 14 __GNU_EH_FRAME
0 OBJECT LOCAL DEFAULT 20 _GLOBAL_OFFSET
2 FUNC GLOBAL DEFAULT 11 __libc_csu_fin
0 NOTYPE WEAK DEFAULT 21 data_start
14 FUNC GLOBAL DEFAULT 11 call2
0 NOTYPE GLOBAL DEFAULT 21 _edata
0 FUNC GLOBAL DEFAULT 12 _fini
0 FUNC GLOBAL DEFAULT UND __libc_start_m
0 NOTYPE GLOBAL DEFAULT 21 __data_start
0 NOTYPE WEAK DEFAULT UND __gmon_start__
0 OBJECT GLOBAL HIDDEN 21 __dso_handle
4 OBJECT GLOBAL DEFAULT 13 _IO_stdin_used
101 FUNC GLOBAL DEFAULT 11 __libc_csu_ini
0 NOTYPE GLOBAL DEFAULT 22 _end
2 FUNC GLOBAL HIDDEN 11 _dl_relocate_s
43 FUNC GLOBAL DEFAULT 11 _start
0 NOTYPE GLOBAL DEFAULT 22 __bss_start
34 FUNC GLOBAL DEFAULT 11 main
0 OBJECT GLOBAL HIDDEN 21 __TMC_END__
23 FUNC GLOBAL DEFAULT 11 call
0 FUNC GLOBAL DEFAULT 10 _init
-playground $
```

```

0x400390 P  _init
0x4003b0 P  _start
0x4003e0 P  _dl_relocate_static_pie
0x4003f0 P  deregister_tm_clones
0x400420 P  register_tm_clones
0x400460 P  __do_global_ctors_aux
0x400490 P  frame_dummy
0x400497 P  call2
0x4004a5 P  call
0x4004bc P  main
0x4004e0 P  __libc_csu_init
0x400550 P  __libc_csu_fini
0x400554 P  _fini
0x400560  _IO_stdin_used
0x600e50 D  __frame_dummy_init_array_entry
0x600e58 D  __do_global_ctors_aux_fini_array
0x600e60  _DYNAMIC
0x600ff8 D  qword
0x601000  _GLOBAL_OFFSET_TABLE_
0x601020  __dso_handle
0x601028  __TMC_END__
0x601040  __libc_start_main
0x601048  __gmon_start__
0x601050  __libc_start_main@@GLIBC_2.2
0x601058  __gmon_start__

```

25 labels

```

; SHF_ALLOC
0000000000601018 db 0x00 ; '.'
0000000000601019 db 0x00 ; '.'
000000000060101a db 0x00 ; '.'
000000000060101b db 0x00 ; '.'
000000000060101c db 0x00 ; '.'
000000000060101d db 0x00 ; '.'
000000000060101e db 0x00 ; '.'
000000000060101f db 0x00 ; '.'
__dso_handle:
0000000000601020 db 0x00 ; '.'
0000000000601021 db 0x00 ; '.'
0000000000601022 db 0x00 ; '.'
0000000000601023 db 0x00 ; '.'
0000000000601024 db 0x00 ; '.'
0000000000601025 db 0x00 ; '.'
0000000000601026 db 0x00 ; '.'
0000000000601027 db 0x00 ; '.'

; Section .bss
; Range: [0x601028; 0x601030] (8 bytes)
; No data on disk
; Flags: 0x3
; SHT_NOBITS
; SHF_WRITE

```

# Symbol Types

\_\_dso\_handle (OBJECT)

Type	Name
P	_init
P	<b>_start</b>
P	_dl_relocate_static_pie
P	deregister_tm_clones
P	register_tm_clones
P	__do_global_ctors_aux
P	frame_dummy
P	call2
P	call
P	main
P	__libc_csu_init
P	__libc_csu_fini
P	_fini
	_IO_stdin_used
D	__frame_dummy_init_array_entry
D	__do_global_ctors_aux_fini_array
	_DYNAMIC
D	qword
	_GLOBAL_OFFSET_TABLE_
	__dso_handle
	__TMC_END__
	__libc_start_main

```

; Section .text
; Range: [0x4003b0; 0x400552] (418 bytes)
; File offset : [944; 1362] (418 bytes)
; Flags: 0x6
;   SHT_PROGBITS
;   SHF_ALLOC
;   SHF_EXECINSTR

; ===== B E G I N N I N G

                                _start:
00000000004003b0      xor     ebp, ebp
00000000004003b2      mov     r9, rdx
00000000004003b5      pop     rsi
00000000004003b6      mov     rdx, rsp
00000000004003b9      and     rsp, 0xf
00000000004003bd      push    rax
00000000004003be      push    rsp
00000000004003bf      mov     r8, __libc_start_main@PLT
00000000004003c6      mov     rcx, __libc_start_main@PLT
00000000004003cd      mov     rdi, main
00000000004003d4      call    qword [0]
00000000004003da      hlt
                                ; endp
00000000004003db      align   32

```

# Symbol Types

\_start (FUNC)

# nm & strings

## SYMBOL LISTER

---

- ▶ Fast and easy way to get symbols
- ▶ Symbol types
  - ▶ B, T, U, w, etc.
- ▶ Addresses

## STRINGS

---

- ▶ Extract strings from a file



GDB is next.